

# Program Reasoning

## 6. First-order Theories

Kihong Heo



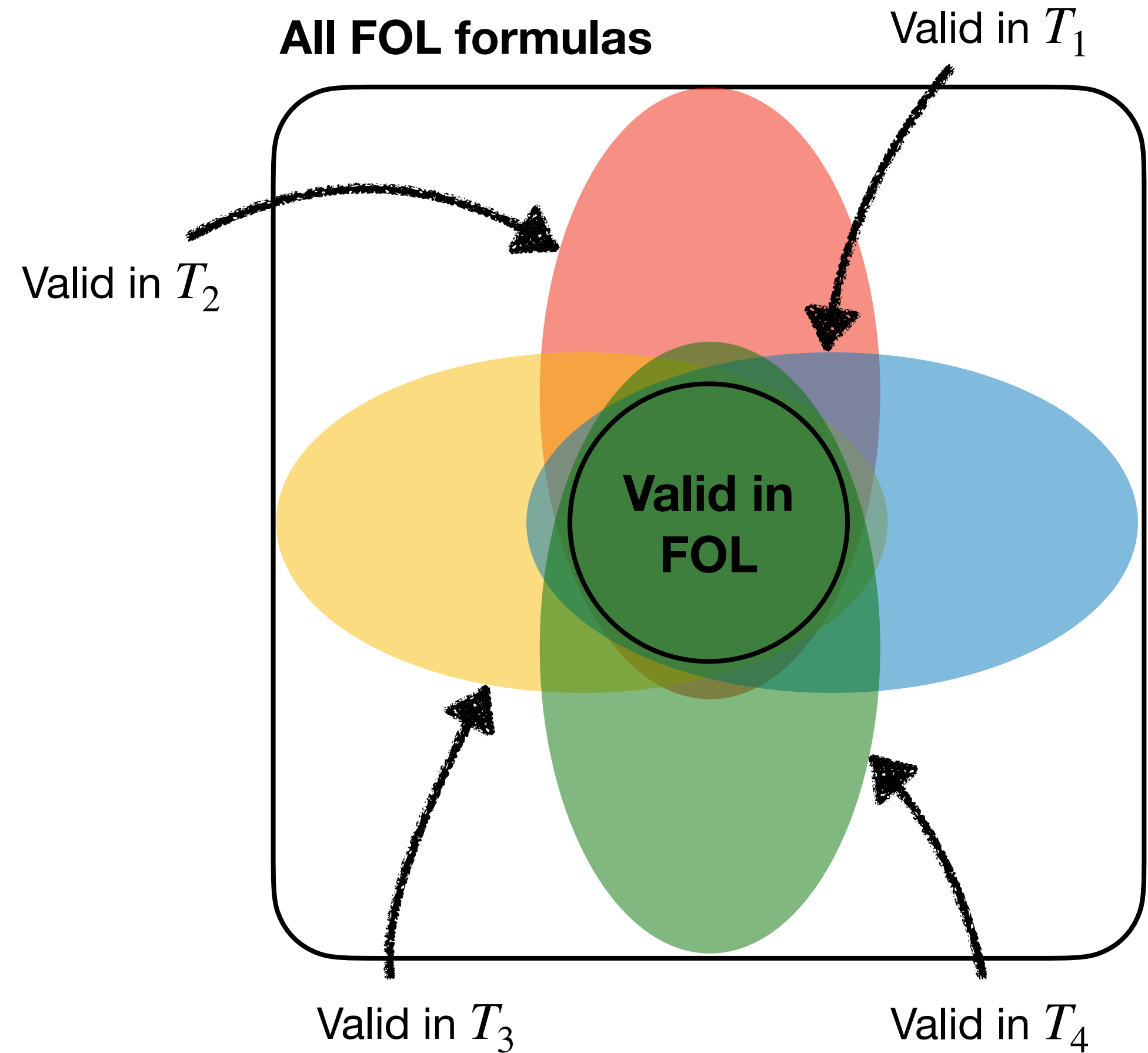
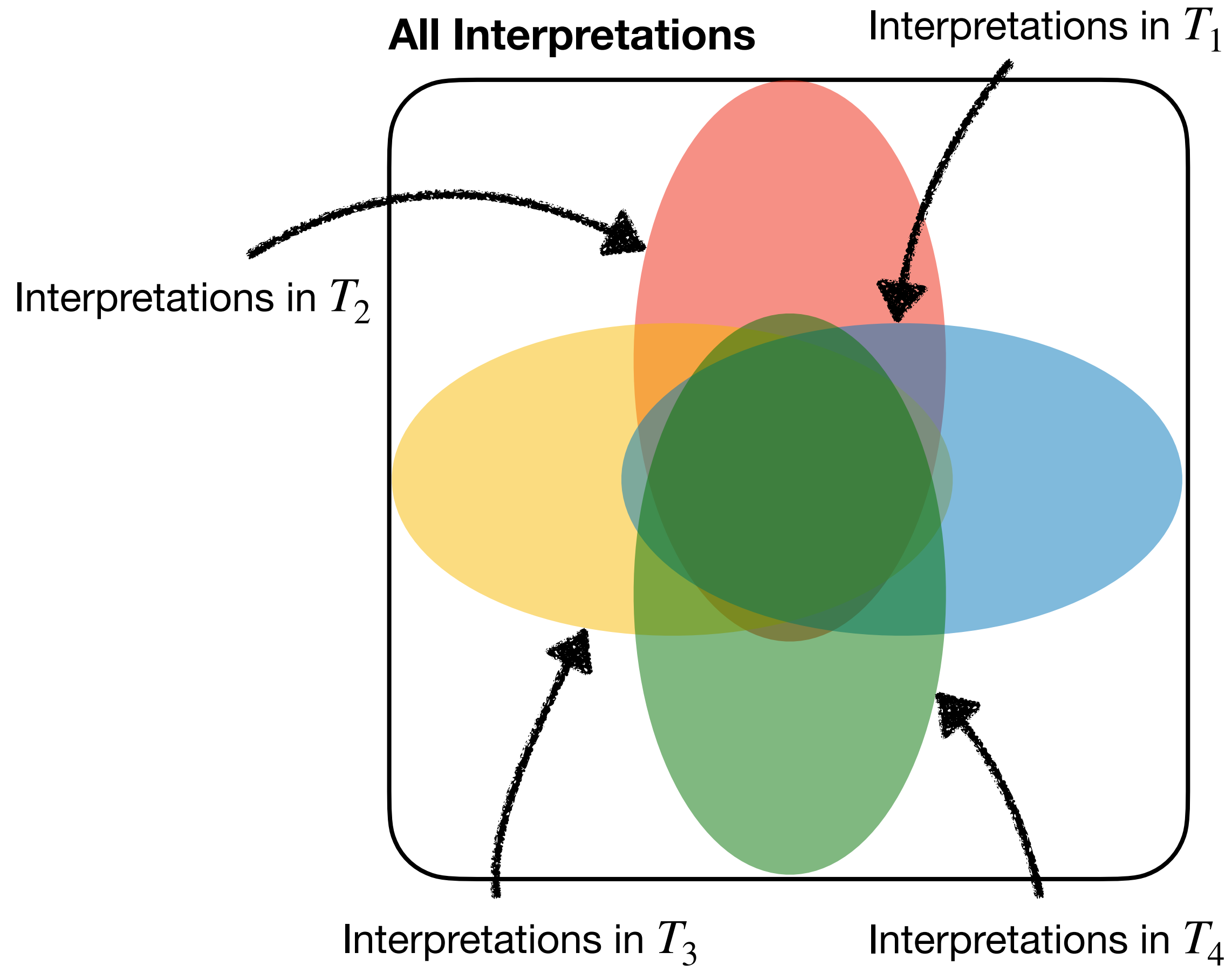
# Motivation (1): Interpretation

- Full first-order logic: functions and predicates are uninterpreted (i.e., determined by  $I$ )
- Validity of full FOL: valid in all interpretations
- Do we really care about all interpretations?
  - For example,  $\forall x . x < x + 1$
- NO. Only **some specific classes (theory)** of interpretations depending on applications
  - Conventional interpretations following axioms
  - E.g., numbers, lists, arrays, strings, etc

# Motivation (2): Decidability

- Validity in FOL: undecidable
- Validity in particular theories: sometimes decidable
- Validity in particular fragments of theories: sometimes decidable or efficiently decidable

# Validity of Theories



# First-order Theory

- Theory  $T$  : A restricted class of FOL
  - Signature  $\Sigma_T$  : a set of constants, functions, and predicate symbols
  - Axioms  $\mathcal{A}_T$  : a set of FOL sentences over  $\Sigma_T$
- $\Sigma_T$ -formula: formula constructed from
  - Symbols of  $\Sigma_T$
  - Variables, logical connectives, and quantifiers
- The symbols of  $\Sigma_T$  does not have prior meaning but the axioms  $\mathcal{A}_T$  provide their meaning

# Theory of Equality $T_E$ (1)

- $\Sigma_E : \{ =, a, b, c, \dots, f, g, h, \dots, p, q, r, \dots \}$
- Equality “=” is an interpreted predicate symbol
  - The conventional interpretation of “=”
  - The meaning is defined via the axioms
- The other functions, predicates, and constants are uninterpreted
- EUF (Equality with Uninterpreted Functions)

# Theory of Equality $T_E$ (2)

- Axioms  $\mathcal{A}_E$ 
  - Reflexivity:  $\forall x . x = x$
  - Symmetry:  $\forall x, y . x = y \rightarrow y = x$
  - Transitivity:  $\forall x, y, z . x = y \wedge y = z \rightarrow x = z$
  - Function congruence:  $\forall \vec{x}, \vec{y} . (\bigwedge_{i=1}^n x_i = y_i) \rightarrow f(\vec{x}) = f(\vec{y})$
  - Predicate congruence:  $\forall \vec{x}, \vec{y} . (\bigwedge_{i=1}^n x_i = y_i) \rightarrow (p(\vec{x}) \leftrightarrow p(\vec{y}))$

# Example (1)

- $D_I = \{0,1\}$
- Which interpretations of  $=$  are allowed in  $T_E$ ?
  - $\alpha_I(=) = \{\langle 0,1 \rangle, \langle 1,0 \rangle\}$
  - $\alpha_I(=) = \{\langle 0,0 \rangle, \langle 1,1 \rangle\}$
  - $\alpha_I(=) = \{\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,0 \rangle, \langle 1,1 \rangle\}$
- Which interpretations of  $f$  are allowed in  $T_E$  when  $\alpha_I(=) = \{\langle 0,0 \rangle, \langle 1,1 \rangle\}$ ?
  - $\alpha_I(f) = \{0 \mapsto 0, 1 \mapsto 1\}$
  - $\alpha_I(f) = \{0 \mapsto 1, 1 \mapsto 0\}$



# Example (2)

- $D_I = \{0,1,2\}$
- Is the following interpretation of  $=$  allowed in  $T_E$ ?
  - $\alpha_I(=) = \{\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,0 \rangle, \langle 1,1 \rangle, \langle 2,2 \rangle\}$
- Which interpretations of  $f$  are allowed in  $T_E$ ?
  - $\alpha_I(f) = \{0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0\}$
  - $\alpha_I(f) = \{0 \mapsto 1, 1 \mapsto 0, 2 \mapsto 2\}$
  - $\alpha_I(f) = \{0 \mapsto 1, 1 \mapsto 2, 2 \mapsto 2\}$

# Validity and Satisfiability Modulo Theory

- $T$ -interpretation: an interpretation that satisfies all the axioms of  $T$ 
  - $I \models A$  for every  $A \in \mathcal{A}$
- $\Sigma_T$ -formula  $F$  is **valid in theory**  $T$  if **all**  $T$ -interpretations satisfy  $F$ 
  - $F$  is  $T$ -**valid** or  $T \models F$
- $\Sigma_T$ -formula  $F$  is **satisfiable in theory**  $T$  if there **exists** a  $T$ -interpretation that satisfies  $F$ 
  - $F$  is  $T$ -**satisfiable**

# Example

- Prove  $F : a = b \wedge b = c \rightarrow g(f(a), b) = g(f(c), a)$  is  $T_E$ -valid

# First-order Theories for Programs

- Equality
- Integers, rationals, and reals
- Lists
- Arrays
- Pointers
- Bit-vectors
- etc

# Theory of Peano Arithmetic (1)

- $\Sigma_{PA} : \{ 0, 1, +, \cdot, = \}$ 
  - 0 and 1 : constants
  - + (addition) and  $\cdot$  (multiplication) are binary functions
  - and = (equality) is a binary predicate

# Theory of Peano Arithmetic (2)

- $\mathcal{A}_{PA}$ : Axioms of  $T_{PA}$ 
  - Zero:  $\forall x . \neg(x + 1 = 0)$
  - Successor:  $\forall x, y . x + 1 = y + 1 \rightarrow x = y$
  - Plus zero:  $\forall x . x + 0 = x$
  - Plus successor:  $\forall x, y . x + (y + 1) = (x + y) + 1$
  - Times zero:  $\forall x . x \cdot 0 = 0$
  - Times successor:  $\forall x, y, z . x \cdot (y + 1) = x \cdot y + x$
  - Induction:  $F[0] \wedge (\forall x . F[x] \rightarrow F[x + 1]) \rightarrow \forall x . F[x]$

An axiom schema for every  $\Sigma_{PA}$ -formula  $F$  with one free variable

# Theory of Peano Arithmetic (3)

- $T_{PA}$ : a powerful theory for arithmetic over natural numbers
- Natural numbers in  $T_{PA}$ 
  - $3x + 5 = 2y$  as  $(1 + 1 + 1) \cdot x + 1 + 1 + 1 + 1 + 1 = (1 + 1) \cdot y$
- Inequality in  $T_{PA}$ 
  - $3x + 5 > 2y$  as  $\exists z. z \neq 0 \wedge 3x + 5 = 2y + z$

# Example (1)

- Prove  $\exists x, y, z. x \neq 0 \wedge y \neq 0 \wedge z \neq 0 \wedge x^2 + y^2 = z^2$  is  $T_{PA}$ -valid



# Example (2)

- Prove  $\forall x, y, z. x \neq 0 \wedge y \neq 0 \wedge z \neq 0 \wedge n > 2 \rightarrow x^n + y^n \neq z^n$  is  $T_{PA}$ -valid

# Theory of Presburger Arithmetic (1)

- $\Sigma_{\mathbb{N}} : \{ 0, 1, +, = \}$ 
  - 0 and 1 : constants
  - + (addition) is a binary function
  - and = (equality) is a binary predicate
- A subset of  $\Sigma_{PA}$  (without multiplication)

# Theory of Presburger Arithmetic (2)

- $\mathcal{A}_{\mathbb{N}}$ : Axioms of  $T_{\mathbb{N}}$ 
  - Zero:  $\forall x . \neg(x + 1 = 0)$
  - Successor:  $\forall x, y . x + 1 = y + 1 \rightarrow x = y$
  - Plus zero:  $\forall x . x + 0 = x$
  - Plus successor:  $\forall x, y . x + (y + 1) = (x + y) + 1$
  - Induction:  $F[0] \wedge (\forall x . F[x] \rightarrow F[x + 1]) \rightarrow \forall x . F[x]$
- A subset of  $\mathcal{A}_{PA}$

An axiom schema for every  $\Sigma_{PA}$ -formula  $F$  with one free variable

# Theory of Lists (1)

- $\Sigma_{cons} : \{ \text{cons}, \text{car}, \text{cdr}, \text{atom}, = \}$ 
  - cons (constructor) is a binary function: “::” in OCaml
  - car (left projector) is a unary function: “List.hd” in OCaml
  - cdr (right projector) is a unary function: “List.tl” in OCaml
  - atom is a unary predicate:  $\text{atom}(x)$  is true iff  $x$  is a single-element list
  - and  $=$  (equality) is a binary predicate

# Theory of Lists (2)

- $\mathcal{A}_{cons}$ : Axioms of  $T_{cons}$ 
  - Reflexivity, symmetry, transitivity of  $T_E$
  - Instantiation of the function congruence for cons, car, and cdr
  - Instantiation of the predicate congruence for atom
  - Left projection:  $\forall x, y . \text{car}(\text{cons}(x, y)) = x$
  - Right projection:  $\forall x, y . \text{cdr}(\text{cons}(x, y)) = y$
  - Construction:  $\forall x . \neg \text{atom}(x) \rightarrow \text{cons}(\text{car}(x), \text{cdr}(x)) = x$
  - Atom:  $\forall x, y . \neg \text{atom}(\text{cons}(x, y))$

# Example

- Prove  $F : \text{car}(a) = \text{car}(b) \wedge \text{cdr}(a) = \text{cdr}(b) \wedge \neg \text{atom}(a) \wedge \neg \text{atom}(b) \rightarrow f(a) = f(b)$  is  $T_{\text{cons}}^-$ -valid

# Theory of Arrays (1)

- $\Sigma_A : \{ \cdot[\cdot], \cdot \langle \cdot \triangleleft \cdot \rangle, = \}$ 
  - $a[i]$  (read) is a binary function: the value of array  $a$  at position  $i$
  - $a \langle i \triangleleft v \rangle$  (write) is a ternary function: the modified array  $a$  in which position  $i$  has value  $v$
  - and  $=$  (equality) is a binary predicate

# Theory of Arrays (2)

- Axioms of  $T_A$ 
  - Reflexivity, symmetry, and transitivity of  $T_E$
  - Array congruence:  $\forall a, i, j. i = j \rightarrow a[i] = a[j]$
  - Read-over-write 1:  $\forall a, v, i, j. i = j \rightarrow a\langle i \triangleleft v \rangle[j] = v$
  - Read-over-write 2:  $\forall a, v, i, j. i \neq j \rightarrow a\langle i \triangleleft v \rangle[j] = a[j]$



# Example

- Prove  $F : a[i] = e \rightarrow \forall j . a\langle i \triangleleft e \rangle[j] = a[j]$  is valid

# Completeness

- A theory  $T$  is **complete** if for every closed  $\Sigma_T$ -formula  $F$ ,  $T \models F$  or  $T \not\models F$ 
  - “We must know, we will know” (David Hilbert)
- What happens if a theory is **incomplete**?
  - “There exists a  $F$  such that we don’t know either  $T \models F$  or  $T \not\models F$ ” (Kurt Gödel)
- Gödel’s 1st incompleteness theorem: “any theory that includes PA is incomplete”
- Example:  $T_{PA}$  is incomplete.

# Consistency

- A theory  $T$  is **consistent** if there is at least one  $T$ -interpretation
- What happens if a theory is **inconsistent**?
  - No interpretation satisfies all the axioms of  $T$  (there exists a contradiction in the axioms)
  - Both  $T \models F$  and  $T \not\models F$ , so  $T \models \perp$
- Example:  $\mathcal{A}_{PA'} = \mathcal{A}_{PA} \cup \{ \forall x . x + 1 = 0 \}$ 
  - Both  $F : 0 + 1 = 0$  and  $\neg F : \neg(0 + 1 = 0)$  are valid
- In a consistent theory  $T$ , there does not exist a  $\Sigma$ -formula  $F$  s.t. both  $T \models F$  and  $T \not\models F$
- Gödel's 2nd incompleteness theorem:  
"Any theory that includes PA cannot prove its own consistency"

# Decidability

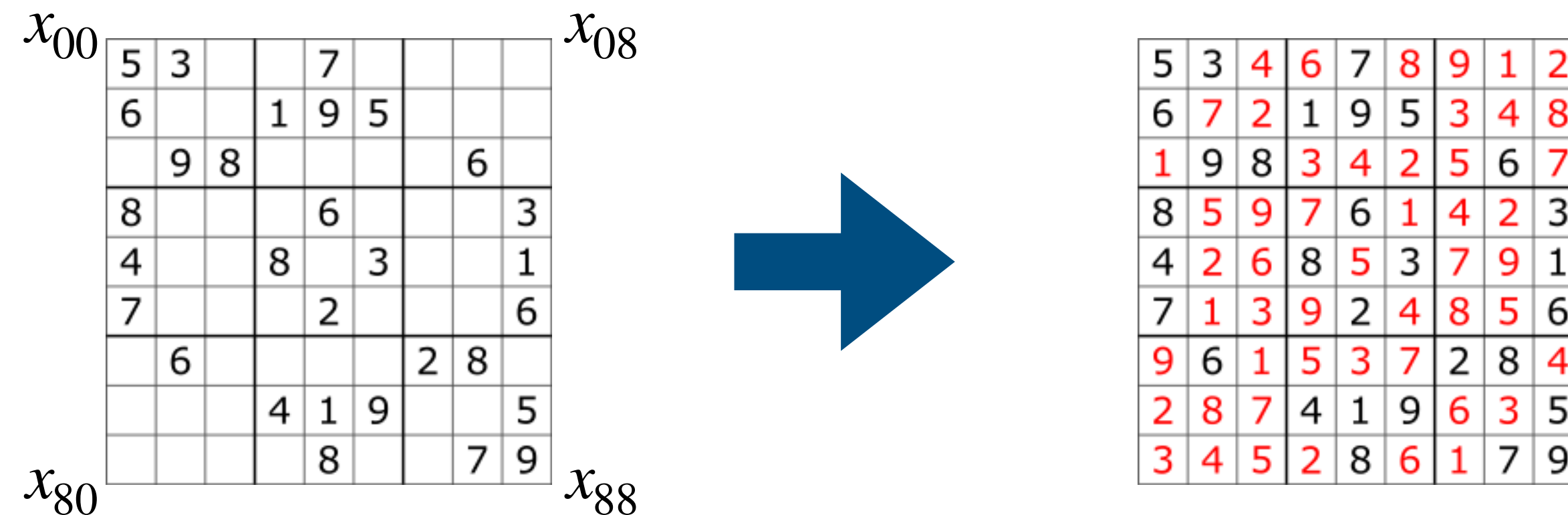
- A theory  $T$  is **decidable** if  $T \models F$  is decidable for every  $\Sigma_T$ -formula  $F$ 
  - Always terminating algorithm
  - Says “yes” if  $F$  is  $T$ -valid, or “no” if  $F$  is  $T$ -invalid
- Many theories are undecidable
  - E.g., the “empty” theory, theory of equality: **undecidable**
- Some theories become **decidable** with further restrictions
  - Quantifier-free fragment: formulae without quantifiers
  - Conjunctive fragment: formulae with only conjunctions

# Decidability of Theories

Description	Full	QFF
equality	no	yes
Peano arithmetic	no	no
Presburger arithmetic	yes	yes
linear integers	yes	yes
reals with multiplication	yes	yes
rational without multiplication	yes	yes
recursive data structures	no	yes
acyclic recursive data structures	yes	yes
arrays	no	yes
arrays with extentionality	no	yes

# Application: Sudoku

- How to solve Sudoku via SMT?



1. Use numbers 1-9:  $\forall 0 \leq i, j \leq 8. 1 \leq x_{ij} \leq 9$
2. Don't repeat any numbers in a row:  $\forall 0 \leq i \leq 8. x_{i0} \neq x_{i1} \neq \dots \neq x_{i8}$
3. Don't repeat any numbers in a column:  $\forall 0 \leq i \leq 8. x_{0i} \neq x_{1i} \neq \dots \neq x_{8i}$
4. Don't repeat any numbers in a square: ...

**Prove  $(1 \wedge 2 \wedge 3 \wedge 4)$  is satisfiable!**

\* <https://en.wikipedia.org/wiki/Sudoku>

# Application: Symbolic Execution

- How to find a crashing input via SMT?

```
void f(int x, int y) {  
    int z = 2 * x;  
    if (y > 0) {  
        int w = 2 * y;  
        if (w + x == 0)  
            crash();  
    }  
}
```

The program crashes if “crash()” is reachable.  
Is this crash possible? What are the values of x and y that cause the crash?

**Prove  $z = 2 \times x \wedge y > 0 \wedge w = 2 * y \wedge w + x = 0$  is satisfiable!**

# Application: Translation Validation (1)

- Compiler bugs



```
$ clang -O0 input.c
$ ./a.out
1
$ clang -O1 input.c
$ ./a.out
Aborted (core dumped)
```



```
# without optimization
$ v8 test.js
true
# with optimization
$ v8 test.js
false
```



# Application: Translation Validation (2)

- How to check the correctness of a compilation via SMT?

# before optimization	# after optimization
f(x) =	f(x) = x
let y = 1 in	
if x = y then 1	
else 1	

The translation is correct if, for all inputs, the return values of  $P_1$  and  $P_2$  are the same

$\iff$  The translation is incorrect if there exists an input such that the return values of  $P_1$  and  $P_2$  are different

1.  $y_{src} = 1 \wedge r_{src} = (\text{if } x_{src} = y_{src} \text{ then } 1 \text{ else } 1)$
2.  $r_{dst} = x_{dst}$
3.  $r_{src} \neq r_{dst}$

**Prove  $(1 \wedge 2 \wedge 3)$  is unsatisfiable!**

# Summary

- First-order theories: instances of FOL
  - Restrict interpretations using axioms
- Many useful theories for program reasoning
  - E.g., equality, integers, arrays, pointers, etc
- Some theories are decidable but some are not
- Many interesting applications
  - E.g., puzzle, bug-finding, verification, etc